
Gruppetime 8

IN1030

Plan for dagen

- Krav
- Kravspesifikasjon
- Kravhåndteringsprosessen

Gruppearbeid: gode ressurser

- For samarbeid: Zoom, Teams, MetroRetro, Google Drive
- For tegning av UML: <https://www.draw.io>,
<https://www.lucidchart.com/pages/>, Figma

Læringsmål

anvende metoder og teknikker for **kravhåndtering**:

- innhente, analysere og spesifisere krav.
- skille mellom funksjonelle og ikke-funksjonelle krav.
- skrive kravspesifikasjon.
- utføre modellering av UML-diagram basert på krav (begynner på dette neste uke).

Litt fra forrige time

Systemutvikling (software engineering)

- Definert som prosessen ved å analysere krav, designe, utvikle, teste, og evaluere
- Både teori og praktisk arbeid inngår
- Teori:
 - ◆ Læren om utvikling av programsystemer av høy kvalitet innen gitte tids- og kostnadsrammer
 - ◆ Omhandler alle teorier, metoder og verktøy for profesjonell utvikling av programvare fra start til slutt.
 - ◆ (IN1030)
- Praktisk:
 - ◆ Arbeidet du faktisk gjør for å lage nye systemer
 - ◆ (IN2000)

Nøkkelbegreper

Systemutvikling

Prosessmodeller

Smidig utvikling

Plandrevet utvikling

Reell prosess

Scrum

Kanban

Fossefallsmodellen

Kravanalyse

Funksjonelle- og ikke-funksjonelle krav

Kostnad

Kravhåndtering

Prosjektplanlegging

UML modellering

Kunde

Risikoanalyse

Implementering

Produserteier

Vedlikehold

Testing

Krav

Krav

- IT systemer lages for at vi kan løse problemer/utnytte muligheter
- Krav forteller oss hva som faktisk skal lages
 - ◆ Både høyere (faktisk produkt) og lavere (produktets komponenter) nivå
- Det kan være kostbart å rette på krav etter systemleveranse



Brukerkrav og systemkrav

→ Brukerkrav

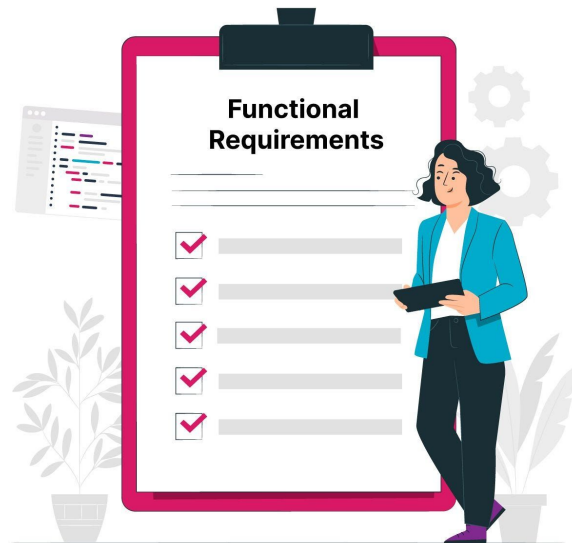
- ◆ Kravene som er uttrykt av kunden eller sluttbrukeren av systemet.
- ◆ Vanligvis formulert i enklere, ikke-tekniske termer som tar sikte på å dekke brukerens behov.

→ Systemkrav

- ◆ Systemkrav er en detaljert og formell beskrivelse av funksjoner, tjenester eller operasjonelle begrensninger som må implementeres i programvaren.
- ◆ Disse kravene er vanligvis skrevet i tekniske termer og spesifiserer nøyaktig hvordan systemet skal fungere.

Funksjonelle krav

- **Hva** systemet skal gjøre.
- **Funksjonelt brukerkrav:** Systemet skal tilby brukere å laste opp bilder
- **Funksjonelt systemkrav:** Systemet bør tillate brukere å laste opp både bilder i jpg og png
- Skrives ofte slik: "Systemet skal.../Systemet bør (nice to have)..."



Funksjonelle krav: eksempler fra IN2000

FUNKSJONELLE KRAV		
MÅ HA	VIL HA	GREIT Å HA
1.1: Brukeren kan plotte inn starttidspunkt og destinasjon	2.1: Punktvis pop-up cards med værinformasjon på ruten	3.1: Informasjon om MC treff
1.2: Tidsbasert værmelding spredd over ruten	2.2: Bensinstasjoner på ruten	3.2: Bytte fargetema (dark mode)
1.3: Filtrering på brukerinput	2.3: Live bilder av føre på ruten	3.3: Melde fartskontroll og promillekontroll
1.4: Legge til mellomdestinasjoner	2.4: Legge til favorittdestinasjoner	3.4: Lagre ruten (cache)
1.5: Ruten vises på et kart	2.5: Kjente landemerker på ruten	3.5: Oppdatere informasjonen med en "refresh"-knapp
1.6: MetAlerts-varslinger		
1.7: Live GPS		

Ruter som eksempel

→ Finn noen funksjonelle krav til Ruter-appen.



Ikke-funksjonelle krav

Krav som ikke direkte beskriver funksjonene som leveres til sluttbruker, men karakteristikker med systemet som helhet; **Hvordan** skal det fungere?

Sier noe om **kvalitetsattributter** systemet skal ha

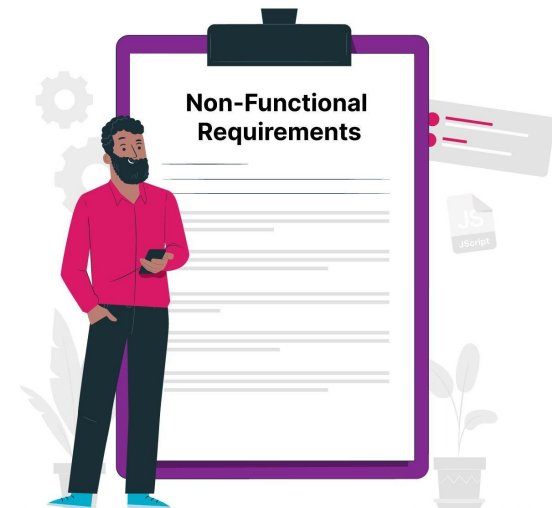
Sier noe om egenskaper - hvordan systemet skal oppføre seg

Bør være **målbare** (mulighet for å teste dem)

Sier noe om:

- Implementasjon
- Reliabilitet
- Responstid
- Sikkerhet
- Skalerbarhet

Mer kritiske → Møter man ikke disse kan det føre til at hele systemet ikke fungerer.



Ikke-funksjonelle krav

Produktkrav: Brukskvalitet/brukervennlighet, ytelse og effektivitet samt lagringsplass, pålitelighet og lagring av data.

Eks: Systemet skal kun bruke 5 MB lagringsplass på bilder.

Organisatoriske krav: kostnader & ressurser, leveransetidspunkt & prosess; utviklingsmodeller, programmeringsspråk, verktøy og komponenter samt generelle standarder og regler.

Eks: Systemets ulike funksjoner skal testes hver andre måned.

Systemets kodehåndtering skal gjøres i GitHub.

Eksterne krav: lovverk, begrensninger & etiske problemstillinger.

Eks: Systemet skal tilfredsstillte 39 av 61 WCAG-retningslinjer.

Ikke-funksjonelle krav: eksempler fra IN2000

IKKE-FUNKSJONELLE KRAV
1.1: Bruke Jetpack Compose Elementer
1.2: Bruke MapBox API for kart
1.3: Bruke MET API for vær
1.4: Værmeldingen oppdateres hvis det har gått mer enn 5 minutter
1.5: Universell utforming (WCAG 2.1-standarden)
1.6: Brukervennlig og intuitivt
1.7: Appen skal ha MVVM-arkitektur
1.8: Lav kobling, høy kohesjon
1.9: UI-elementer skal være lett å gjenbruke

Ruter som eksempel

→ Finn noen ikke-funksjonelle krav til Ruter-appen.



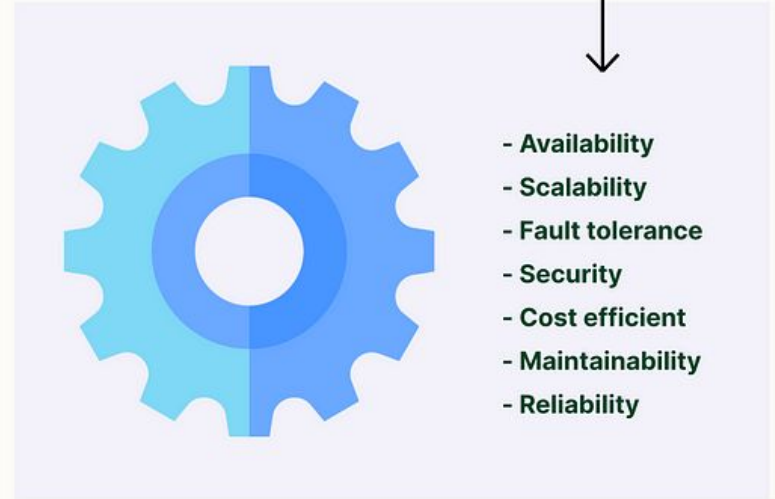


USER

FUNCTIONAL REQUIREMENTS



NON-FUNCTIONAL REQUIREMENTS



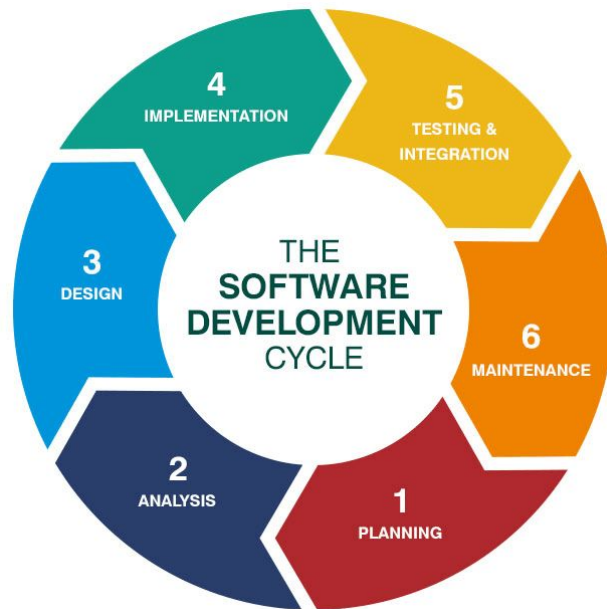
- Availability
- Scalability
- Fault tolerance
- Security
- Cost efficient
- Maintainability
- Reliability

THE SYSTEM

Kravhåndtering

Kravhåndtering

- Kravhåndtering er prosessen å identifisere, analysere og spesifisere kravene til det nye eller forbedrede systemet



Kravhåndteringsprosessen

→ Steg 1 - forstudie/målanalyse:

- ◆ Analysere nå-situasjonen i et case, og ønsket situasjon og mulig tiltak for å oppnå ønsket situasjon

→ Steg 2 - kravinnsamling og kravanalyse:

- ◆ Hva ønsker interessentene seg (behov) → prioritering av kravene (must have, should have...)

→ Steg 3 - kravspesifisering (kravspec):

- ◆ Lage enighet mellom kunde/aktør og leverandør for hva systemet skal levere

→ Steg 4 - validering av kravspec:

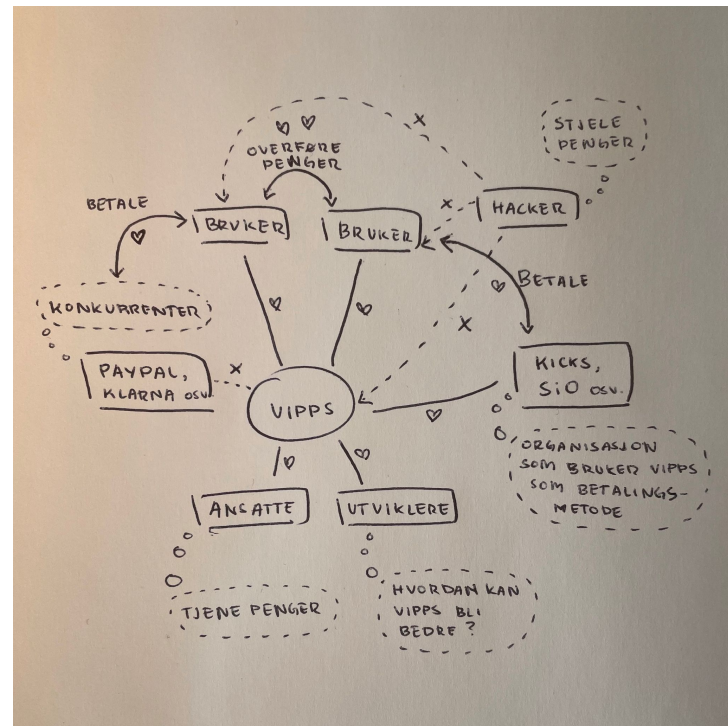
- ◆ Sjekke om vi møter kunde/aktørens behov. Eventuelle mangler/problemer?

→ Steg 5 - håndtering av kravspec:

- ◆ Nye behov fra brukere etter systemets implementasjon? Gjøre endringer?

Steg 1 - Forstudie

- Kartlegge hvem som er aktør/interessent?
- Hvilke funksjoner ønsker de?
- Politiske, økonomiske eller alternative aktørers krav?
- Eksempel -> rike bilder



Steg 2 - Kravinnsamling og kravanalyse

- Samle inn data om hvilke krav og behov interessentene har (f.eks. Intervju, spørreskjema osv.)
- Analysere denne dataen og danne krav
- Prioritere kravene

FUNKSJONELLE KRAV

MÅ HA	VIL HA	GREIT Å HA
1.1: Brukeren kan plote inn starttidspunkt og destinasjon	2.1: Punktvis pop-up cards med værinformasjon på ruten	3.1: Informasjon om MC treff
1.2: Tidsbasert værmelding spredd over ruten	2.2: Bensinstasjoner på ruten	3.2: Bytte fargetema (dark mode)
1.3: Filtrering på brukerinput	2.3: Live bilder av føre på ruten	3.3: Melde fartskontroll og promillekontroll
1.4: Legge til mellomdestinasjoner	2.4: Legge til favorittdestinasjoner	3.4: Lagre ruten (cache)
1.5: Ruten vises på et kart	2.5: Kjente landemerker på ruten	3.5: Oppdatere informasjonen med en "refresh"-knapp
1.6: MetAlerts-varslinger		
1.7: Live GPS		

IKKE-FUNKSJONELLE KRAV

1.1: Bruke Jetpack Compose Elementer
1.2: Bruke MapBox API for kart
1.3: Bruke MET API for vær
1.4: Værmeldingen oppdateres hvis det har gått mer enn 5 minutter
1.5: Universell utforming (WCAG 2.1-standarden)
1.6: Brukervennlig og intuitivt
1.7: Appen skal ha MVVM-arkitektur
1.8: Lav kobling, høy kohesjon
1.9: UI-elementer skal være lett å gjenbruke

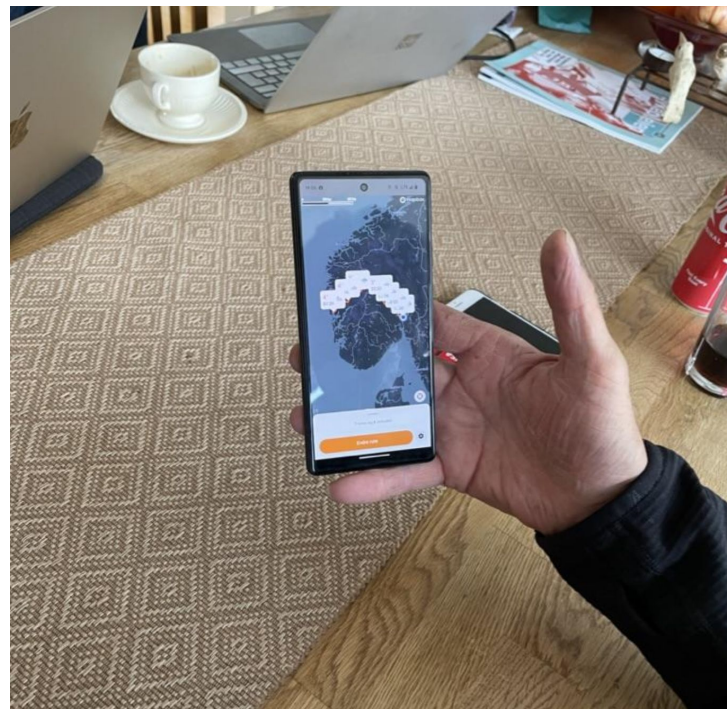
Steg 3 - Kravspesifikasjon (kravspec)

- Skaper felles forståelse av systemet og enighet om som skal leveres.
- Er grunnlag for kontrakt som viser hva leverandør og kunde blir enige om.
- Forhindrer eventuelle konflikter som kan oppstå på bakgrunn av uklare forventninger.



Steg 4 - Validering av kravspec

- Møter vi aktørens faktiske behov?
- Finne eventuelle problem/mangler med kravspesifikasjonen.
- Kan gjøres gjennom brukertesting.



Steg 5 - Kravendringer

- Når forståelsen av et problem endrer seg
 - ◆ Plutselig får sluttbruker et endret perspektiv på problemet systemet skal løse
- Håndtering av kravendringer
 - ◆ Hva skal prioriteres? Tid, penger, arbeidskraft...
 - ◆ Ha en plan for hvordan dette kan håndteres



Nøkkelbegreper

Systemutvikling

Prosessmodeller

Smidig utvikling

Plandrevet utvikling

Reell prosess

Scrum

Kanban

Fossefallsmodellen

Kravanalyse

Funksjonelle- og ikke-funksjonelle krav

Kostnad

Kravhåndtering

Prosjektplanlegging

UML modellering

Kunde

Risikoanalyse

Implementering

Produserteier

Vedlikehold

Testing

Spørsmål?

Ikke nøl med å sende oss en mail:)

erikaesk@uio.no

sandefo@uio.no
